



马永杰工作日报

2025-02-11

2025-02-10

1. 观看周例会

2. 结果分析

	中-英	泰-英	英-印地	印地-英	英-印尼
趋势最相近	Comet 和 BLEURT	BLEURT 和 xCOMET	COMET 和 xCOMET	BLEURT 和 COMET	COMET 和 xCOMET
趋势最不相近	TRE 和 ChatGPT error stat	TRE 和 Prompt Template	TRE 和 ChatGPT error stat	TRE 和 Prompt Template	TRE 和 ChatGPT error stat
最相近的指标对通常涉及 COMET、BLEURT 和 xComet，这表明这些基于神经网络的评估指标有较强的一致性					
最不相近的指标对通常涉及 TRE 与其他指标，特别是与 ChatGPT error stat 和 Prompt Template（更有可能考虑到评估了不同的特征）					

- 最稳定的指标对（在所有语言对中保持高相关性）
 - COMET 与 xComet（中文到英文: 0.91，泰语到英文: 0.88，英文到印地语: 0.94，印地语到英文: 0.90，英文到印尼语: 0.93）
 - BLEURT 与 COMET（中文到英文: 0.92，泰语到英文: 0.87，英文到印地语: 0.89，印地语到英文: 0.91，英文到印尼语: 0.90）
- 最不稳定的指标对（相关性波动大）
 - TRE 与 ChatGPT error stat（中文到英文: 0.21，泰语到英文: 0.25，英文到印地语: 0.18，印地语到英文: 0.24，英文到印尼语: 0.20）

评论（4）

>>

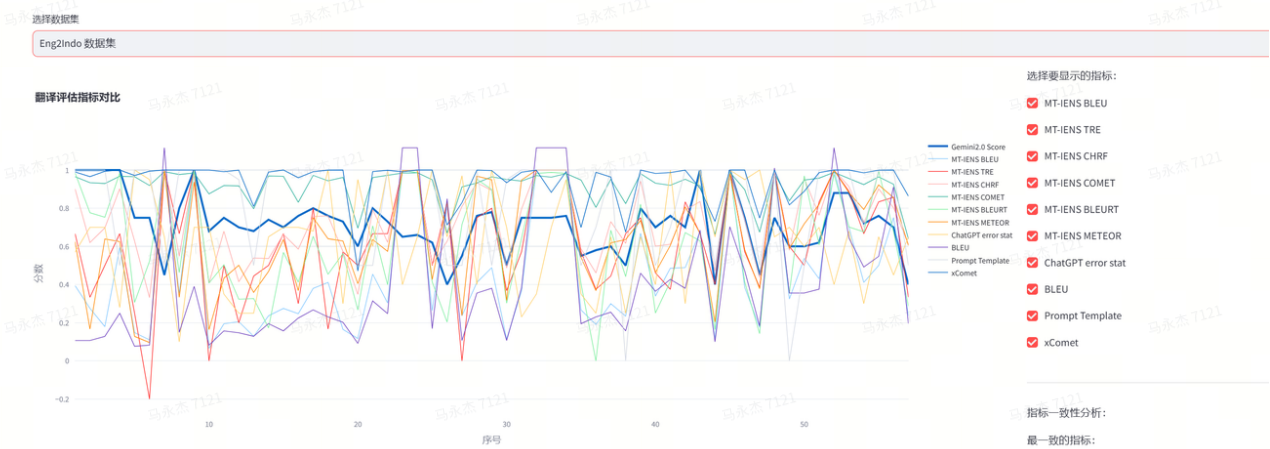
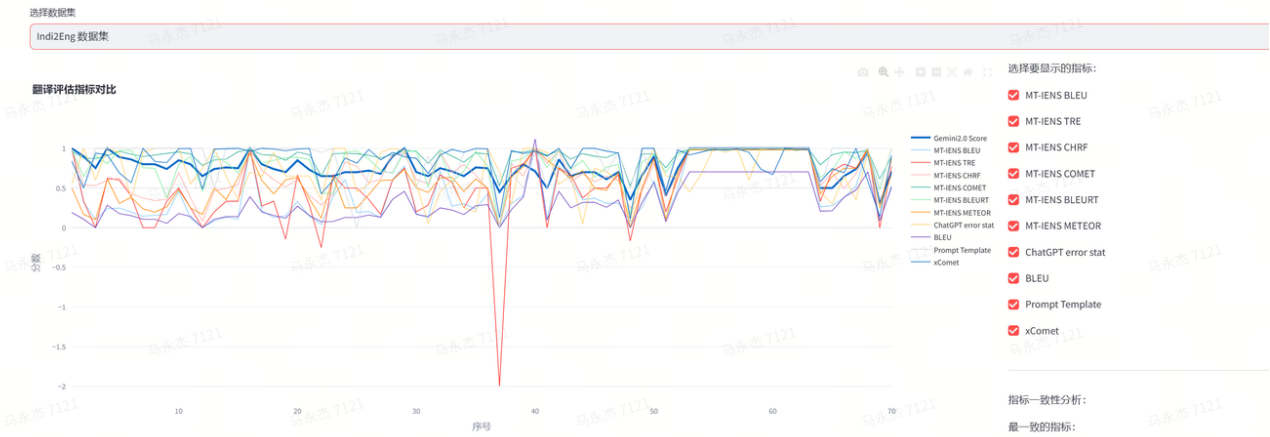
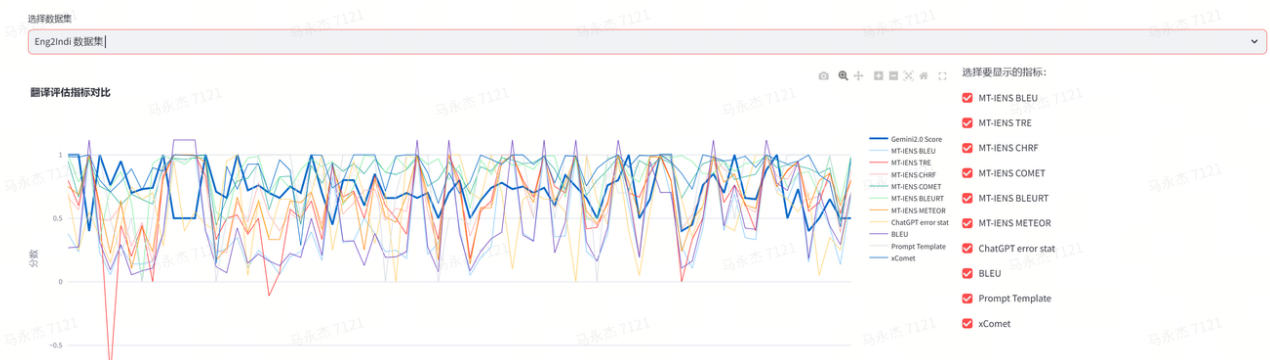
- 基于相关性可以将指标分为三大类
 - COMET ,BLEURT, xCOMET 组内高相关，跨语言稳定性好
 - BLEU, METEOR, CHRF 中等相关,不同语言对表现差异较大
 - TRE, ChatGPT error stat, Prompt Template 低相关，独立性强
- 实际使用建议
 - COMET 或 xComet 作为主要评估指标（跨语言稳定性好，与其他指标相关性合理）
 - 互补指标搭配
 - COMET + TRE（覆盖不同维度）
 - BLEURT + ChatGPT error stat（结合统计和人工评估）
 - 从语言角度
 - 中英翻译：重点关注COMET和BLEURT
 - 泰英翻译：结合BLEURT和xComet
 - 印地语相关：以COMET为主，配合TRE
 - 印尼语相关：COMET和xComet搭配使用

3. 代码收集上传

2025-02-08

补全五个数据集的指标分析
汇总其他同事的数据





2025-02-07

- 对中译英，英译泰，英译印地三个数据集进行不同指标的测试，BLEU，CHRF，TER，METEOR，ROUGE1，ROUGE2，ROUGEL，COMET，BLEURT 包含基于重叠参考和基于神经网络参考的指标。
- 将所有指标的量纲统一到[0,1]，趋势一致，便于比较
- 从不同角度比较分析，如

指标相关性分析摘要

最相关的指标对:

ROUGE1 和 ROUGEL

相关系数: 0.949

👉 这两个指标具有很强的相关性，可能在评估相似的翻译特征。

最不相关的指标对:

BLEU 和 BLEURT

相关系数: 0.581

👉 这两个指标可能在评估不同的翻译特征，组合使用可能更全面。

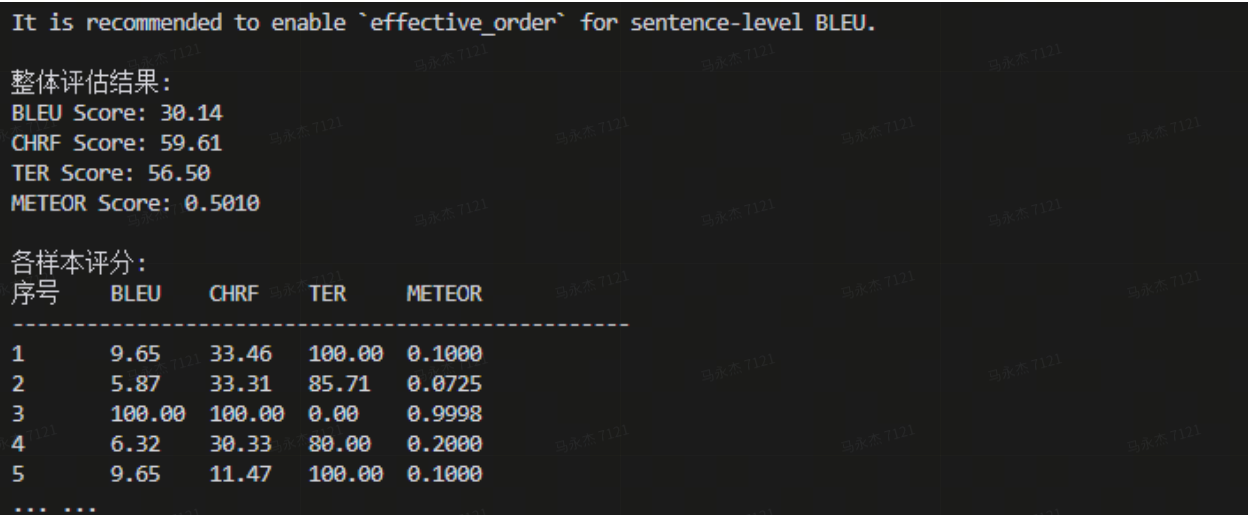
在不同的数据集上指标的相关性表现并不一致

- 在实际评估中，可选择相关性较低的指标组合，以获得更全面的评估
- ...

2025-02-06

asr_result	ast_result	gemini_result
唯心者何正?	What is right for idealists?	How can idealists be correct?
圣阿基兰德人口变化图示。	Saint Archiland population and telepho	Chart showing population changes in St. Aq
沱江镇自清朝开始就一直凤凰县的行政中	Tuojiang Town has been the administr	Tuojiang Town has been the administrative
该市镇年时的	The town in that year	This town, in the year...

共100条测试样本



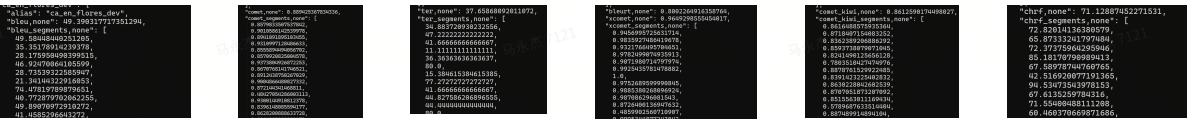
计算结果

- BLEU Score: 30.14
- CHRF Score: 59.61
- TER Score: 56.50
- METEOR Score: 0.5010

以上为指标平均值（还需补充其他指标），对于每个样本的不同指标的差异可绘制折线图等进行比较。

2025-02-05

4. MT-LENS支持的指标有BLUE，TRE, CHRF, COMET等常见指标，如截图所示，主要以bleu来评估。集成现有的指标用于多维度的评估。



5. CATER 没有给出相应的源码，API无法调用

2025-01-24

- 在mt-evaluation框架下，CTranslate 模型能够顺利运行，其预训练中-英翻译模型部分翻译如：
CTranslate2 中——>英
圣阿基兰德人口变化图示：St. Aquillant's demographic change map.
不是让你拿来玩的：It's not for you to play with.























这类物质通常是通过让靶细胞因裂解而溶解而起效的：This type of material usually works by dissolving target cells by cracking.

在中国鸦片泛滥的年代，不同材质的烟枪甚至成为了身份和地位的象征：In China's **opium-rich** years, smoke guns of different materials have even become symbols of identity and status

2. 记录CTranslate评估错误：

ValueError: Unable to open SentencePiece model ./ctranslate2/opus-mt-en-ca-ctranslate2/spm.model

Huggingface 上并没有 spm.model文件，报错需要该文件。尝试其他模型

gaudi	model.bin updated to accomodate ctranslate 4.4.0	05d8fc1	3 months ago	
.gitattributes	 Safe	387 Bytes 	README.md Update	6 months ago
README.md	 Safe	6.72 kB 	README.md Update	6 months ago
config.json	 Safe	215 Bytes 	Initial Commit	6 months ago
generation_config.json	 Safe	293 Bytes 	updated permissions	6 months ago
model.bin	 LFS	156 MB 	model.bin updated to accomodate ctranslate 4.4_	3 months ago
shared_vocabulary.json	 Safe	1.3 MB 	updated permissions	6 months ago
source.spm	 Safe	805 kB  LFS 	README.md Update	6 months ago
target.spm	 Safe	807 kB  LFS 	README.md Update	6 months ago
tokenizer_config.json	 Safe	44 Bytes 	updated permissions	6 months ago
vocab.json	 Safe	1.62 MB 	updated permissions	6 months ago

2025-01-23

1. 团队内讨论

2. 测试：

- MT-LENS: An all-in-one Toolkit for Better Machine Translation Evaluation (<https://arxiv.org/pdf/2412.11615>)
- CATER: Leveraging LLM to Pioneer a Multidimensional, Reference Independent Paradigm in Translation Quality Evaluation(<https://arxiv.org/pdf/2412.11261>)
 - 调用API太慢，目前没有给出具体的模型

2025-01-22

1. 论文调研

1.1 How Good Are GPT Models at Machine Translation? A Comprehensive Evaluation(<https://arxiv.org/pdf/2302.09210>)

- GPT模型翻译与机器翻译的对比
 - 句子级的评估：COMET-22
 - 文档级的评估：改进的COMET
 - COMET 是句子级别的评估指标，专注于单个句子的翻译质量
 - 语义嵌入：使用预训练的语言模型（如BERT）将源文本、机器翻译输出和参考翻译转换为语义嵌入向量。

- **相似性计算：**
 - 计算源文本与机器翻译输出之间的语义相似度。
 - 计算机器翻译输出与参考翻译之间的语义相似度。
- **评分生成：**
 - 根据相似度计算结果，生成一个综合评分，反映机器翻译输出的质量。
- **Modified COMET：将文档以滑动窗口的形式分割成不同片段，在每个片段内分别评估，计算平均得分**
 - 允许在其上下文中评估每个句子。
 - 由于滑动窗口的重叠性质，允许在多个上下文中评估每个句子。
 - 避免了评估模型的有限上下文窗口，这可能会阻碍文档上较长静态窗口的质量评估。
- 人工评估

1.2 Awesome LLM MT

- [Is ChatGPT A Good Translator? Yes With GPT-4 As The Engine](#). Wenxiang Jiao, Wenxuan Wang, Jen-tse Huang, Xing Wang, Zhaopeng Tu. (arxiv 2023) {code}
- [A Multitask, Multilingual, Multimodal Evaluation of ChatGPT on Reasoning, Hallucination, and Interactivity](#). Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, Quyet V. Do, Yan Xu, Pascale Fung. (arxiv 2023) {code}
- [How Good Are GPT Models at Machine Translation? A Comprehensive Evaluation](#). Amr Hendy, Mohamed Abdelrehim, Amr Sharaf, Vikas Raunak, Mohamed Gabr, Hitokazu Matsushita, Young Jin Kim, Mohamed Afify, Hany Hassan Awadalla. (arxiv 2023) {code}
- [Document-Level Machine Translation with Large Language Models](#). Longyue Wang, Chenyang Lyu, Tianbo Ji, Zhirui Zhang, Dian Yu, Shuming Shi, Zhaopeng Tu. (arxiv 2023) {code}
- [Multilingual Machine Translation with Large Language Models: Empirical Results and Analysis](#). Wenhao Zhu, Hongyi Liu, Qingxiu Dong, Jingjing Xu, Shujian Huang, Lingpeng Kong, Jiajun Chen, Lei Li. (arxiv 2023) {code}
- [How to Design Translation Prompts for ChatGPT: An Empirical Study](#). Yuan Gao, Ruili Wang, Feng Hou. (arxiv 2023)
- [Investigating the Translation Performance of a Large Multilingual Language Model: the Case of BLOOM](#). Rachel Bawden, François Yvon. (EAMT 2023) {code}
- [Large language models effectively leverage document-level context for literary translation, but critical errors persist](#). Marzena Karpinska, Mohit Iyyer. (arxiv 2023) {code}
- [Do GPTs Produce Less Literal Translations?](#). Vikas Raunak, Arul Menezes, Matt Post, Hany Hassan Awadalla. (ACL 2023)
- [Zeno GPT-MT Report](#). Graham Neubig. (github 2023) {code}
- [ChatGPT MT: Competitive for High- \(but not Low-\) Resource Languages](#). Nathaniel R. Robinson, Perez Ogayo, David R. Mortensen, Graham Neubig. (WMT 2023)

多为使用BLEU等指标的方法

1.3 The Assessment of Professional Translation Quality: Creating Credibility out of Chaos (<https://core.ac.uk/download/pdf/59325137.pdf>)

- 说明性文章，未提出具体的方法，

1.4 Multilingual Machine Translation with Large Language Models: Empirical Results and Analysis (<https://arxiv.org/pdf/2304.04675>)

- 哪些因素会影响 LLM 的翻译表现？
 - 多资源
 - 好的翻译模板，如 【x】 可以翻译为 【y】
 -都是通过BLEU来验证佐证这些结论

1.5 Document-Level Machine Translation with Large Language Models (<https://arxiv.org/pdf/2304.02210>)

- 语境感知指标
 - CTT 术语翻译的一致性

$$CTT = \frac{\sum_{t \in \mathbf{TT}} \frac{\sum_{i=1}^k \sum_{j=i+1}^k \mathbb{1}(t_i = t_j)}{C_k^2}}{|\mathbf{TT}|}$$

- AZPT 代词翻译的准确性

$$AZPT = \frac{\sum_{z \in \mathbf{ZP}} A(t_z | z)}{|\mathbf{ZP}|}$$

2025-01-21

1. CATER的Scoring Methodology具体计算过程如下:

1.1 流程

- i. 对于每个评估维度(LA, SA, CF, STA, IC),都会识别出相关的错误,并给出以下信息:
 - 错误位置(引用有问题的片段)
 - 错误解释(为什么属于该类别的错误)
 - 修正建议
 - 需要修改的词数

- ii. 对于每个评估维度,总结以下信息:
 - 检测到的总错误数
 - 需要修改的总词数
 - 错误类型(如语法错误、遗漏等)

- iii. 计算每个维度的编辑比率(Edit Ratio, ER%):
 $ER\%_{category} = (Words\ to\ Correct_category / Original\ Word\ Count) * 100$

- iv. 将每个维度的ER%转换为0-100的Category Score:
 $Score_category = \max\{0, \lceil (1 - (ER\%/100 * Weight)) * 100 \rceil\}$ 其中Weight为每个维度的权重,可以根据需求进行调整。如果某个维度没有错误,则ER%=0.0%,Score=100。

- v. 计算Overall Score:
 $Overall\ Score = (LA_Score + SA_Score + CF_Score + STA_Score + IC_Score) - 400$ 如果结果为负数,则Overall Score设为0。如果所有维度都满分(100),则Overall Score为100。

- vi. 计算Overall ER%:

Overall ER%为各个维度ER%之和。

总之,CATER的Scoring Methodology通过识别错误、计算编辑比率和分数,为翻译质量提供了多维度、可解释的评估结果,有助于针对性地改进翻译质量。

1.2 测试链接 <https://erudaite.ai/cater>

2. 团队内讨论

- 补充论文

3. 论文调研

3.1 综述 Translation Quality Assessment: A Brief Survey on Manual and Automatic Methods (<https://arxiv.org/abs/2105.03311>)

- 自动检测
 - 基于单词匹配的方法（速度快，缺乏考虑句法等其他信息）
 - 单词顺序
 - 序列匹配距离
 - 精度和召回率
 - 基于深度特征方法
 - 句法相似
 - 语义相似
 - 基于神经网络的方法

3.2 Quality-Aware Decoding for Neural Machine Translation (<https://arxiv.org/pdf/2205.00978>)

- 动机：能否利用 MT 质量评估的最新进展来生成更好的翻译？如果是这样，如何才能最有效地做到这一点
- 质量感知解码框架
 - 生成候选翻译
 - 无参考/有参考的MT指标衡量
 - 对其进行排名，并选择最高排名对应的候选翻译作为最终翻译
- 结论：与基于 MAP 的解码相比，质量感知解码根据强大的自动评估指标和人工判断，可以带来更好的翻译。

3.3 Tradition and Trends in Translation Quality Assessment (https://www.researchgate.net/profile/Jitka-Zehnalova/publication/294260655_Tradition_and_Trends_in_Translation_Quality_assessment/links/56bf7b9f08ae2f498ef7f692/Tradition-and-Trends-in-Translation-Quality-assessment.pdf)

- 概述了传统和当前的翻译质量评估(TQA)方法，发表于2013年，概述论文新颖性不够
- 提出了一个处理TQA问题的宏观三级模型

3.4 ...

1. 翻译评估资料

1.1 CATER: Leveraging LLM to Pioneer a Multidimensional, Reference Independent Paradigm in Translation Quality Evaluation(<https://arxiv.org/pdf/2412.11261>)

- 五个方面评估
 - 语言准确性 (Linguistic Accuracy, LA)
 - 语义准确性 (Semantic Accuracy, SA)
 - 上下文契合度 (Contextual Fit, CF)
 - 文体适当性 (Stylistic Appropriateness, STA)
 - 信息完整性 (Information Completeness, IC)
- 不需要参考译文

1.2 MT-LENS: An all-in-one Toolkit for Better Machine Translation Evaluation (<https://arxiv.org/pdf/2412.11615>)

- MT-LENS支持多种评估任务,包括翻译质量、性别偏差检测、以及对拼写错误的鲁棒性,并提供一个平台。
- 可选指标
 - 基于重叠的参考指标:BLEU、TER、CHRF 等
 - 基于神经网络的参考指标:COMET、BLEURT、METRICX 等
 - 质量估计指标:METRICX-QE、COMET-KIWI 等
 - 包含错误跨度的指标:XCOMET、XCOMET-QE 等
 - 基于词表的指标:ETOX
 - 基于嵌入的指标:MUTOX、DETOXIFY 等

1.3 Evaluating the Translation Performance of Large Language Models Based on Euas-20 (<https://arxiv.org/pdf/2408.03119>)

- 数据集: Euas-20 (ChatGPT生成了1000个中文句子,然后使用Google Translate将其翻译成其他目标语言,构建了这个多语言数据集)
- 评价指标 Bleu (传统) 和 COMET (<https://arxiv.org/pdf/2009.09025>)
- 结果显示,尽管大语言模型的翻译性能有所提高,其中Llama-3表现尤为出色,但这些模型在不同语言上的翻译能力仍然非常不平衡。特别是在处理低资源语言时,它们仍然面临巨大挑战。
- 高质量和多样化的语料库在提高大语言模型的翻译性能方面起到了重要作用

1.4 MMTE: Corpus and Metrics for Evaluating Machine Translation Quality of Metaphorical Language

- 四个方面 (隐喻、情感、真实性和质量)
- 评估隐喻质量

1.5 Results of WMT22 Metrics Shared Task: Stop Using BLEU– Neural Metrics Are Better and More Robust (<https://aclanthology.org/2022.wmt-1.2.pdf>)

- 展示了神经网络学习指标相对于像 BLEU、SPBLEU 或 CHRF 这样的重叠指标的优越性

1.6 Towards Explainable Evaluation Metrics for Machine Translation (<https://www.jmlr.org/papers/volume25/22-0416/22-0416.pdf>)

- 当前倾向于使用较为简单的指标如BLEU。作者认为这可能是由于缺乏对新指标的透明性和可解释性

1.7 A Method for Evaluating English Translation Quality of Intelligent Translation Robots based on Long Short-Term Memory

- 能够处理长序列的单词，考虑句子上下文，并在语言间词序不同的情况下准确翻译以保持意义

1.8 Quality Evaluation Model of Automatic Machine Translation based on Deep Learning Algorithm

2025-01-17

- 1 容器内外部数据交互
- 2 `docker cp /path/to/local/file <container_id>:/path/in/container`
- 3 `docker cp <container_id>:/path/to/file/on/container /path/to/destination/on/host`
- 4 如果只是一次性复制文件，使用`docker cp`。
- 5 如果需要持续同步文件，使用卷

收集翻译评估资料

[Towards explainable evaluation metrics for machine translation](#)

<https://www.jmlr.org/papers/volume25/22-0416/22-0416.pdf>

Evaluating the Translation Performance of Large Language Models Based on Euas-20

<https://arxiv.org/pdf/2408.03119>

MT-LENS: An all-in-one Toolkit for Better Machine Translation Evaluation

<https://arxiv.org/pdf/2412.11615>

Results of WMT22 Metrics Shared Task: Stop Using BLEU– Neural Metrics Are Better and More Robust

<https://aclanthology.org/2022.wmt-1.2.pdf>

CATER: Leveraging LLM to Pioneer a Multidimensional, Reference-Independent Paradigm in Translation Quality Evaluation

<https://modelscope.cn/papers/100413/summary>

MMTE: Corpus and Metrics for Evaluating Machine Translation Quality of Metaphorical Language

<https://modelscope.cn/papers/91576>

A Method for Evaluating English Translation Quality of Intelligent Translation Robots based on Long Short-Term Memory

<https://ieeexplore.ieee.org/abstract/document/10594107>

Quality Evaluation Model of Automatic Machine Translation based on Deep Learning Algorithm

<https://ieeexplore.ieee.org/document/10169967>

...

2025-01-16

1. rsync

- `rsync -av /source/directory /destination/directory` 本地同步
- `rsync -av -e ssh /source/directory user@remote:/destination/directory` 远程同步
- `-a` 归档模式，综合选项(相当于选择了其他所有可选参数-rlptgoD)
- `-v` 详细模式，显示传输过程
- `-e ssh` 使用SSH进行加密传输
- 往服务器推送数据或者从服务器下拉数据都在本地执行

2. git相关命令

- `mkdir demo_repository -> cd demo_repository -> git init` 创建一个目录并初始化仓库
- 初始化仓库后默认有且仅有一个名为“master”的分支，可对其改名
- 创建并切换到新分支
- `git checkout -b new_branch` `-b`的作用就是若不存在就创建并切换，不加`-b`，若不存在便会切换失败
- `git checkout existed_branch`
- `git branch` 列出所有分支
- `git branch new_branch` 创建新分支，与checkout类似
- `git branch -d existed_branch` 删除分支

- `git cherry-pick commit_hash` 有两个分支1和2，当在1中提交了若干内容，但是想将提交内容中的部分提交到2中，此时只需要记住1中提交的部分内容的hash值，并切换到分支2，使用该命令就可以将1中的部分迁移到2中，来源于短语cherry-pick 意为择优挑选，形象！
- `git rebase main` 将当前分支变基到main分支上（将main中从与当前分支不一致的地方到最新的提交这一段复制到当前分支）
- `git rebase -i HEAD~3` 交互式变基
- `git reset --hard commit_hash` 重置到特定提交，丢弃所有更改
- `git reset --soft commit_hash` 重置到特定提交，保留更改在工作目录
- `git tag v1.0` 创建标签
- `git tag` 查看标签
- `git tag -d v1.0` 删除标签
- `git merge branch` 合并该分支到当前分支
- `git diff` 查看未暂存的更改
- `git diff --cached` 查看已经暂存的更改

3. Docker 相关命令

3.1 镜像管理

- `docker images` 列出本地所有镜像
- `docker pull image` 从docker hub下载镜像
- `docker build -t image path` 从 Dockerfile 构建镜像
- `docker rmi image` 删除本地镜像

3.2 容器管理

- `docker ps` 列出正在运行的容器
- `docker ps -a` 列出所有容器
- `docker run image` 运行容器
- `docker stop container_id` 停止运行的容器(id)
- `docker start container_id` 启动停止的容器(id)
- `docker restart container_id` 重启容器
- `docker rm container_id` 删除容器
- `docker exec -it container_id /bin/bash` 进入容器

3.3 网络管理

- `docker network ls` 列出所有网络
- `docker network create <network>` 创建一个新的网络
- `docker network rm <network>` 删除一个新的网络

3.4 卷管理

- `docker volume ls` 列出所有卷
- `docker volume create <volume>` 创建新的卷
- `docker volume rm <volume>` 删除某个卷

3.5 信息和状态

- docker info docker系统信息
- Docker version 版本信息

3.6 日志和监控

- Docker logs container_id 查看日志内容
- Docker stats 显示所有容器的资源使用情况

4. 翻译测评理论调研

4.1 贝叶斯定理

4.1.1 先验概率

先验概率是指在没有观察到任何数据之前，对某个事件发生的概率的估计。在翻译中，先验概率反映了模型在**未考虑**上下文或其他信息的情况下，对单词或短语翻译的初步判断。

4.1.2 后验概率

后验概率是在观察到某些数据后，对事件发生概率的更新。在翻译中，后验概率**考虑了**上下文信息、语法结构和语义关系等因素，从而调整对翻译结果的判断。

4.1.3 概率推断

4.1.3.1 初始翻译（先验概率）

大模型可能首先基于单词到单词的映射（例如，基于训练数据中“i” m”对应“我是”，“lucky”对应“幸运的”，“dog”对应“狗”）生成初步翻译。这一过程可以视为对单词的直接翻译，反映了模型的先验知识。

4.1.3.2 矫正翻译（后验概率）

- 要求模型进行矫正并考虑上下文时，模型会利用后验概率来重新评估翻译结果。在这个过程中，模型会考虑以下几个方面：
- **语义一致性**：模型会识别“狗”作为一个名词，而“我”作为主语，通常情况下，人不会被称为“狗”。因此，模型会调整翻译以确保语义上的一致性。
- **上下文信息**：模型可能会利用上下文信息（如句子中其他词汇的含义和语法结构）来判断“lucky dog”在特定上下文中的更自然的翻译。
- **条件概率**：当主语是“我”时，模型会计算在这种情况下“lucky dog”翻译为“幸运儿”的条件概率，并得出更合适的翻译结果。

4.1.4 数学表达

$$P(A|B) = (P(B|A) \times P(A))/P(B)$$

- A 是翻译结果（我是一个幸运儿/ 我是一个狗）
- B是观察到的输入（i'm a lucky dog）
- P(A)是先验概率，表示翻译为“我是一个幸运儿”的初步估计
- $P(B|A)$ 是似然概率，表示在已知翻译为“我是一个幸运儿”的情况下，输入“i' m a lucky dog”出现的概率。
- $P(B)$ 是边际概率，表示输入“i' m a lucky dog”出现的总概率。

- $P(A|B) < (P(B|A) \times P(A))/P(B)$

2025-01-15

1. 模型推理

i. GPU内存不足

- 指定空闲GPU(或多个GPU)

```
1 device = "cuda:4"
2 torch.cuda.set_device(device)
```

- 清理缓存

```
1 torch.cuda.empty_cache()
```

- 设置环境变量

PYTORCH_CUDA_ALLOC_CONF=expandable_segments:True 避免碎片化

```
1 os.environ["PYTORCH_CUDA_ALLOC_CONF"] =
  "expandable_segments:True"
```

- 调整最大tokens(256-->128-->64)
- 输入调整 (image = image.resize((256, 256)))

ii. 内存消耗

- 输入张量内存消耗+模型参数内存消耗+其他（激活、优化器等）
- 参数量：8291375616

```
1 total_params = sum(p.numel() for p in model.parameters())
2 print(f"Total parameters: {total_params}")
```

- 类型：半精度float16（已改ed）

```
1 for name, param in model.named_parameters():
2     print(f"Layer: {name} | Size: {param.size()} | Type:
    {param.dtype}")
```

- $8291375616 \times 2 = 16,582,751,232 / 1024 / 1024 = 15816\text{MB} = \mathbf{15.44\text{GB}}$
Error: torch.OutOfMemoryError: CUDA out of memory. **Tried to allocate 12.20 GiB.** GPU 4 has a total capacity of 23.65 GiB of which 6.43 GiB is free. Process 122240 has 17.21 GiB memory in use. Of the allocated memory 16.72 GiB is allocated by PyTorch, and 36.60 MiB is reserved by PyTorch but unallocated. If reserved but unallocated memory is large try setting PYTORCH_CUDA_ALLOC_CONF=expandable_segments:True to avoid fragmentation. See documentation for Memory Management
- **15.44+12 > 24 一张跑不起来**

```
1 import os
2 import requests
3 from PIL import Image
4 from io import BytesIO
5 from transformers import
  Qwen2VLForConditionalGeneration, AutoTokenizer,
  AutoProcessor
6 from qwen_vl_utils import process_vision_info
7 import torch
8
9
10 os.environ["PYTORCH_CUDA_ALLOC_CONF"] =
  "expandable_segments:True"
11
12
13 device = "cuda:4"
14 torch.cuda.set_device(device)
15
16
17 image = image.resize((256, 256))
18 image_folder = "/vllm-workspace/images"
19 os.makedirs(image_folder, exist_ok=True)
20
21
22 url = "https://qianwen-res.oss-cn-
  beijing.aliyuncs.com/Qwen-VL/assets/demo.jpeg"
23 response = requests.get(url)
24 image = Image.open(BytesIO(response.content))
25
26
27 local_image_path = os.path.join(image_folder,
  "demo.jpeg")
28 image.save(local_image_path)
29
30
31 messages = [
32     {
33         "role": "user",
34         "content": [
35             {"type": "image", "image": local_image_path},
36             {"type": "text", "text": "Describe this
  image."},
37         ],
38     }
39 ]
40
```

```

41 model =
    Qwen2VLForConditionalGeneration.from_pretrained("/app/mod
    el", torch_dtype=torch.float16, device_map={"": device})
42 total_params = sum(p.numel() for p in model.parameters())
43 print(f"Total parameters: {total_params}")
44
45 for name, param in model.named_parameters():
46     print(f"Layer: {name} | Size: {param.size()} | Type:
        {param.dtype}")
47
48 processor = AutoProcessor.from_pretrained("/app/model")
49
50 text = processor.apply_chat_template(
51     messages, tokenize=False, add_generation_prompt=True
52 )
53 image_inputs, video_inputs =
    process_vision_info(messages)
54 inputs = processor(
55     text=[text],
56     images=image_inputs,
57     videos=video_inputs,
58     padding=True,
59     return_tensors="pt",
60 )
61 inputs = inputs.to(device)
62 print(f"Input tensor shape:
    {inputs['pixel_values'].shape}")
63
64 summary = torch.cuda.memory_summary(device=None,
    abbreviated=True)
65 print(summary)
66
67 torch.cuda.empty_cache()
68
69 generated_ids = model.generate(**inputs,
    max_new_tokens=64)
70 generated_ids_trimmed = [
71     out_ids[len(in_ids):] for in_ids, out_ids in
    zip(inputs.input_ids, generated_ids)
72 ]
73 output_text = processor.batch_decode(
74     generated_ids_trimmed, skip_special_tokens=True,
    clean_up_tokenization_spaces=False
75 )
76 print(output_text)
77

```

2. 模型结构

```

1 Qwen2VLModel(
2     (embed_tokens): Embedding(152064, 3584)
3     (layers): ModuleList(
4         (0-27): 28 x Qwen2VLDecoderLayer(
5             (self_attn): Qwen2VLSdpaAttention(
6                 (q_proj): Linear(in_features=3584, out_features=3584,
                    bias=True)
7                 (k_proj): Linear(in_features=3584, out_features=512,
                    bias=True)

```



```

8      (v_proj): Linear(in_features=3584, out_features=512,
      bias=True)
9      (o_proj): Linear(in_features=3584, out_features=3584,
      bias=False)
10     (rotary_emb): Qwen2VLRotaryEmbedding()
11     )
12     (mlp): Qwen2MLP(
13         (gate_proj): Linear(in_features=3584,
      out_features=18944, bias=False)
14         (up_proj): Linear(in_features=3584,
      out_features=18944, bias=False)
15         (down_proj): Linear(in_features=18944,
      out_features=3584, bias=False)
16         (act_fn): SiLU()
17     )
18     (input_layernorm): Qwen2RMSNorm((3584,), eps=1e-06)
19     (post_attention_layernorm): Qwen2RMSNorm((3584,), eps=1e-
      06)
20 )
21 )
22 (norm): Qwen2RMSNorm((3584,), eps=1e-06)
23 (rotary_emb): Qwen2VLRotaryEmbedding()
24 )
25

```

3. 总结

- 模型：Qwen/Qwen2-VL-7B-Instruct
(<https://huggingface.co/Qwen/Qwen2-VL-7B-Instruct>)
- 镜像：vllm/vllm-openai (<https://hub.docker.com/r/vllm/vllm-openai>)
- 下载：检查准备工具（git / modelscope 等）
 - 关于遇到的git 下载的大文件大小不一致的问题，原因是对于大文件，使用git lfs 时大文件会被替换为指针文件，这也是为什么小文件的SHA256码一致，但大文件出错的原因。（在此耗费了大量时间，分析问题不够细致，并没有意识到这不是错误，而是版本控制问题）
- docker镜像
 - 启动容器，并将本地的模型以及脚本文件挂载到容器内（此处有一个概念理解出现偏差，对于docker的临时性是推出就自动删除，但后面发现并不是退出就自动删除，而是还继续存在，可以再次进入或者删除。
- 重要步骤
 - 相关库导入 os, requests, Qwen2VLForConditionalGeneration, AutoProcessor等
 - 指定显卡
 - b. export CUDA_VISIBLE_DEVICES=4,5,...：指定要使用的显卡，6，7为对应序号
 - c. echo \$CUDA_VISIBLE_DEVICES：验证指定显卡成功与否，输出对应序号即可
 - d. 指定后，在使用时，编号映射到0，1，如果不以这种方式，可以直接使用具体逻辑设备ID号4，5
 - 图像处理，进行缩放裁剪，可减少内存消耗

- 模型加载：使用 `Qwen2VLForConditionalGeneration` 和 `AutoProcessor` 加载模型和处理器，并将模型放置于 CUDA 设备上
- 通过 `processor` 将文本和图像输入处理为模型所需的格式，并确保所有输入张量都在同一设备上
- 可以改进的地方，对于GPT的使用需更加规范，提高问题分析的准确性。

2025-01-14

1. docker镜像下载

- i. 下载命令：`docker Pull [-----]`
- ii. 下载问题解决
 - `docker pull vllm/vllm-openai` （网络问题）
 - 检查docker运行状态：`systemctl status docker`
 - 配置镜像（https://yeasy.gitbook.io/docker_practice/install/mirror）

2. 模型部署

i. 基本逻辑

```
docker run --runtime=nvidia --gpus=all -it （或gpus=""device=0,1""）
-v /data/Qwen2-VL-7B-InstructMYJ:/app/model （本地模型挂载到容器内）
-v /data/Qwen2-VL-7B-InstructMYJ/infer.py:/app/script/infer.py （本地文件
挂载到容器内）
--entrypoint /bin/bash(覆盖镜像的默认入口点，使得容器在启动后进入
Bash shell,便于手动操作)
vllm/vllm-openai （镜像名称）
```

```
1 docker run --runtime=nvidia --gpus=all -it \
2 -v /data/Qwen2-VL-7B-InstructMYJ:/app/model \
3 -v /data/Qwen2-VL-7B-
  InstructMYJ/infer.py:/app/script/infer.py \
4 --entrypoint /bin/bash \
5 vllm/vllm-openai
```

ii. demo

- 下载qwen_vl_utils (太慢...)
- 镜像下载（`pip install qwen-vl-utils -i https://pypi.tuna.tsinghua.edu.cn/simple`）

```
1 from transformers import
  Qwen2VLForConditionalGeneration, AutoTokenizer,
  AutoProcessor
2 from qwen_vl_utils import process_vision_info
3
4 model =
  Qwen2VLForConditionalGeneration.from_pretrained("/app/mod
  el", torch_dtype="auto", device_map="auto")
```

（网络问题）



李靖瑜 1月14日 19:33
电脑上登录飞连，可以解决访问
境外依赖被墙问题。



马永杰 1月14日 19:54
好滴！

```

5
6
7 processor = AutoProcessor.from_pretrained("Qwen/Qwen2-VL-
  7B-Instruct")
8
9
10 messages = [
11     {"role": "user",
12      "content": [
13          {"type": "image", "image": " https://qianwen-
  res.oss-cn-beijing.aliyuncs.com/Qwen-VL/assets/demo.jpeg
  "},
14      ],
15      {"type": "text", "text": "Describe this
  image."}},
16     ],
17     }
18 ]
19
20 text = processor.apply_chat_template(
21     messages, tokenize=False, add_generation_prompt=True
22 )
23 image_inputs, video_inputs =
  process_vision_info(messages)
24 inputs = processor(
25     text=[text],
26     images=image_inputs,
27     videos=video_inputs,
28     padding=True,
29     return_tensors="pt",
30 )
31 inputs = inputs.to("cuda")
32
33 generated_ids = model.generate(**inputs,
  max_new_tokens=128)
34 generated_ids_trimmed = [
35     out_ids[len(in_ids) :] for in_ids, out_ids in
  zip(inputs.input_ids, generated_ids)
36 ]
37 output_text = processor.batch_decode(
38     generated_ids_trimmed, skip_special_tokens=True,
  clean_up_tokenization_spaces=False
39 )
40 print(output_text)

```

- 第4行代码，参数1此处的路径要与docker run 时的一致，这是一个从本地模型路径到容器内的路径映射（本地:容器）



docker镜像里，可能没有模型，需要启动docker镜像的时候，做一个服务环境文件目录到镜像内部的文件目录的映射(from Huangyi)

- 内存不足。。。

iii. 相关命令

- docker ps 查看正在运行的容器
- docker exec -it <容器_ID> /bin/bash 正在运行的容器
- ps aux | grep mayongjie 筛选进程

3. Ubuntu安装

- i. Vmware 虽然运行在一个独立的虚拟机中，与主机系统完全隔离，带来额外的资源消耗，性能不如直接在主机上运行
- ii. wsl: 直接在windows内核运行，开销小；与windows的集成好，方便文件和工具的共享
- iii. <https://www.bing.com/videos/riverview/relatedvideo?q=windows+%e7%9b%b4+%e6%8e%a5+%e8%a3%85ubuntu&mid=C24DCD024884083B841C24DCCD024884083B841&FORM=VAMTRV>

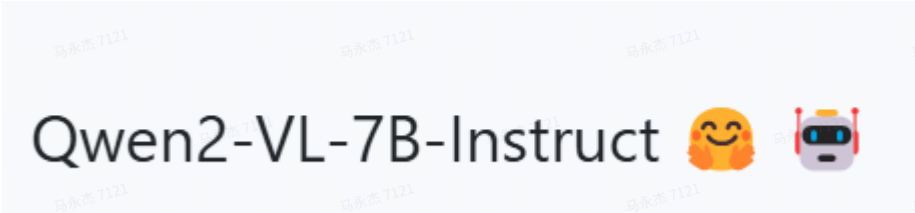
4. 模型推理

- i. docker(包含或不包含对应模型)提供基本的环境
- ii. 对应的模型
- iii. 本地模型路径到容器内部模型路径的链接(映射)
- iv. 本地文件路径到容器内部文件路径的链接(映射)
- v. 推理便是根据已有模型在docker环境下不断优化文件内容，达到目标需求的过程

2025-01-13

1. 模型下载

- a. 下载途径（HuggingFace和ModelScope）



左HuggingFace 右ModelScope

- i. HuggingFace
 - Https
 - SSH（还未搞懂下载的模型与远程仓库中文件大小不一致的原因，但相关配置正确，多次尝试仍出现该问题，后续需搞懂...）
- ii. ModelScope
 - 下载ModelScope后，根据网页提示即可
- b. 下载问题
 - i. 明确下载路径
 - 下载模型到指定路径 `modelscope download --model 'Qwen/Qwen2-7b' --local_dir 'path/to/dir'`
 - ii. 下载前检查磁盘状态（df -h），防止中途磁盘空间溢出导致下载中止
- c. 问题：占了根盘，已解决

2. Docker

- a. Image 可执行文件/副本
- b. Container 集装箱，运行起来的进程

像



黄毅 1月14日 19:09

docker 镜像一般不会放模型 开源模型都是通过一个服务镜像推理的 没有必要这么做



马永杰 1月14日 19:23

好的，这个弄的时候，印象里记得好像说过，就回去翻了一下聊天记录

达到



黄毅 1月14日 19:11

这条不理解表达的什么意思



马永杰 1月14日 19:25

就是原来模型可能是适用于普通话，但二次开发做方言的模型时，可能需要不断优化提示

c. 仓库

d. 所有文件-->进行“编译打包”（docker build)-->生成“可执行的文件” 镜像 image
运行镜像（docker run） -->容器

2025-01-07

1. 提示工程

- 引导模型

1.1 配置参数

- a. 输出长度
- b. 温度：随机性 高温--高随机性；低温--低随机性
- c. Top-K采样：限制选择范围为前K个最高概率token，平衡确定性和随机性
- d. Top-P采样：累计概率选择
- e. ...

1.2 提示技术

2. 服务器模型部署

- i. 常用命令

1. Nvidia-smi :显示所有显卡状态
2. export CUDA_VISIBLE_DEVICES=6,7,...：指定要使用的显卡，6，7为对应序号
3. echo \$CUDA_VISIBLE_DEVICES：验证指定显卡成功与否，输出对应序号即可

2.1 Models 下载

1. Git
2. Git lfs

2.2 Docker 镜像

2025-01-06

1. 大模型推理

1.1 相关概念

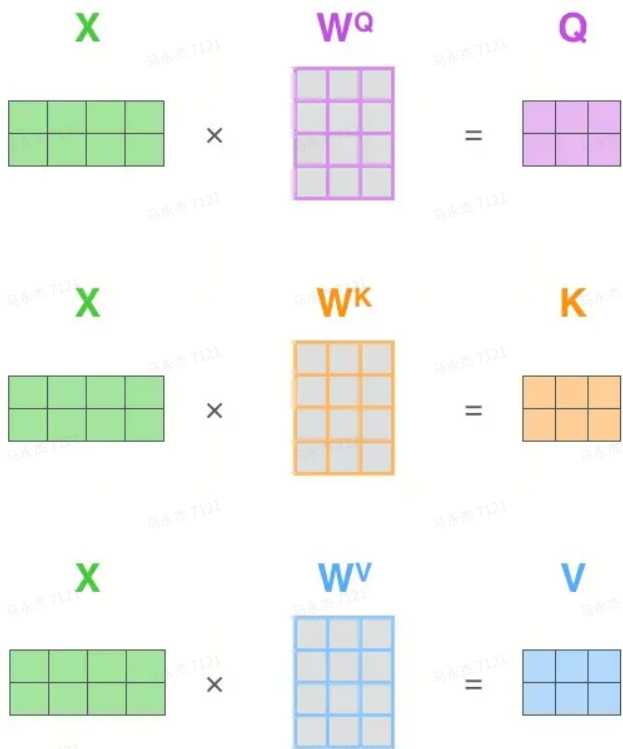
- 吞吐量：可并发处理的任务量
- 延时：处理单个任务的时间
- 分词(token)：处理的基本单元，不仅限于单个单词的能独立表示含义的语
句片段

如“dubug”分为“de”（减少）和“bug”（漏洞）两个token，而非一个；对于“decrease”，“debug”，模型只需记住“bug”，“crease”，“de”三个意思即可知道“bug”，“crease”，“debug”，“decrease”4个词的意思，基于“de”还可以推导出“decode”（解码），“devaluation”（贬值）等词。

1.2 注意力计算过程

$$\text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Q, K, V三个矩阵都是从输入矩阵X线性变换而来:



$$Q = XW^Q$$

$$K = XW^K$$

$$V = XW^V$$

W^Q, W^K, W^V 是三个可训练优化的参数，X与三者相乘后相当于经历一次线性变换。对注意力进行扩展，即增加多组优化矩阵，每组分别计算生成不同的Q, K, V矩阵。**即多头注意力**

1.3 KV Cache

在计算注意力过程中，存在重复计算，所以通过缓存前一步的计算结果来降低每次的计算成本（空间换时间）。按顺序只计算 $\sqrt{}$ 位置，其他位置覆盖不计算。

分词(token)：处理的基本单元，不仅限…



李靖瑜 1月6日 19:30（编辑过）

给你一些评论供参考哦

【关于 Token 划分】

在大模型的分词机制中(如BERT的WordPiece，和GPT的BPE)，“debug”通常不会被拆成“de”和“bug”，而是被视为一个整体或依其他统计方式拆分。

例如，BERT/WordPiece可能会将“unhappiness”分成“un”，“##happi”，“##ness”，并不是基于前缀或后缀的直观拆分。

【关于词义推理】

虽然基于“de-”等前缀可以做初步推断，但在大模型的语言处理里，上下文和海量样本的统计关联才是关键。换言之，不能仅凭“de”表示“减少”就简单推断所有带“de-”的单词含义。

【改进建议】

1.建议多了解分词原理（如BERT的WordPiece、GPT的BPE等），看看它们如何在大量语料中自动学习最优子词。

2.进一步学习语义分析和上下文

✓				
✓	✓			
✓	✓	✓		
✓	✓	✓	✓	
✓	✓	✓	✓	✓

1.4 批处理

- Naive Batchinig
- Static Batching
- Continuous Batching

1.5 PagedAttention

- 大-->小;
- 复杂度的降低 $O(n^2)$ --> $O(m^2)$ $m \ll n$;
- 页内，页件分别计算注意力;

1.6 Llama

- Decoder-only
- 基于已经生成的上下文信息计算（注意力矩阵满秩）

2025-01-03

1. 数据预处理与清洗

- a. 基本流程（文本层面）：
 - i. 解析-->数据尺寸（特征选择/数据舍弃/数据补充等）-->特殊字符过滤(敏感字符、符号、特殊标志、不同语种等)
- b. 语义层面：
 - i. 文件去重（度量不同文件之间相似性，SimhHash）
 - 1. SimHash: 分词->hash映射-->加权-->累加合并-->降维（稀疏化）-->海明距离（ $\sqrt{}$ ）
 - ii. 基于句子，段落去重
 - iii. 语句/段落质量过滤

2. 大模型微调

- a. RLHF（Reinforcement Learning from Human Feedback）
 - i. 预训练语言模型 (LM)/监督微调；
 - ii. 聚合问答数据并训练奖励模型 (Reward Model, RM)；
 - iii. 用强化学习 (RL) 方式微调 LM。
- b. 微调方案：

在词义理解中的重要性，才能更准确地把握大模型处理单词的方式。

3.同时，可在实际项目里多实验分词器对不同单词的切分结果，对比之下会更直观地理解“大模型并不会按前缀后缀来切词”。

[展开](#)

永杰 马永杰 1月6日 19:56

@李靖瑜 明白了，感觉自己理解有点教条化了，感谢瑜哥😊

- i. Freeze（控制变量法）可保留大部分训练知识，成本低；
- ii. P-Tuning：将固定prompt转换到嵌入层，并用简单的模型处理（相当于函数调用）
- iii. LoRA（Low-Rank Adaptation）：用更少的训练参数来近似全参数微调所得的增量参数。通过引入两个低秩矩阵A和B，将原始权重矩阵的 更新表示为这两个矩阵的乘积（即AB），从而大大减少了可训练参数量。
- iv. 全参数微调（最佳）

3. 模型评估方法

4. Transformer

- a. 词嵌入（向量化，512维度）+位置向量
- b. 自注意力机制（已有词之间相关性,norm到[0,1],越->1,越相关）

